Lightweight Gossip-based Distribution Estimation

Amir H. Payberah†, Hanna Kavalionak‡, Alberto Montresor‡, Jim Dowling†, Seif Haridi†
†Swedish Institute of Computer Science (SICS)
‡University of Trento, Italy
†{amir,jim,seif}@sics.se ‡{name.family}@unitn.it

***Abstract*—Monitoring the global state of an overlay network is vital for the self-management of peer-to-peer (P2P) systems. Gossip-based algorithms are a well-known technique that can provide nodes locally with aggregated knowledge about the state of the overlay network. In this paper, we present a gossip-based protocol to estimate the global distribution of attribute values stored across a set of nodes in the system. Our algorithm estimates the distribution both efficiently and accurately. The key contribution of our algorithm is that it has substantially lower overhead than existing distribution estimation algorithms. We evaluated our system in simulation, and compared it against the state-of-the-art solutions. The results show similar accuracy to its counterparts, but with a communication overhead of an order of magnitude lower than them.**

I. Introduction

Monitoring the components of distributed systems is necessary if they are to become self-managing. One potential solution to provide an accurate view of the system state is to employ a central node that communicates with all individual nodes and monitors their state. Although this approach can provide the desired global knowledge, it is not always feasible or desirable in large-scale dynamic distributed systems, e.g., peer-to-peer (P2P) systems, due to excessive overhead on the central node.

Gossip-based aggregation algorithms [1] are a common solution to this problem. The general structure of gossip-based aggregation algorithms is as follows: each node periodically communicates with a randomly selected partner taken from a peer sampling service [2], [3] and both nodes subsequently exchange local information about the global state of the system. Within a small number of iterations of the algorithm, each node's local estimation converges to a good estimation of the global state.

The first aggregation protocols to appear only provided a single point estimate of the global attribute value, e.g., the network size or the global average of the local attribute values at all nodes [1]. However, practical systems often require an estimate of the distribution of attributes values. For example, CLive [4] needs an estimate of the upload bandwidth distribution across all nodes to compute the number of nodes that can be served in a live video streaming service, and LIMOSENSE [5] requires the distribution of various attribute values to detect hardware and software defects or intrusion attempts. To the best of our knowledge, ADAM2 [6] and EQUIDEPTH [7] are the only available gossip-based solutions for the distribution estimation problem.

In this paper, we present a practical gossip-based distribution estimation protocol that has an order of magnitude less overhead than ADAM2 [6] and EQUIDEPTH [7], while obtaining a comparable accuracy.

II. Related work

The estimation of aggregates in large-scale distributed systems has been well-studied in the past [1], [8], [9], [10]. A popular technique is the hierarchical approach [8], [9], [11], [12], where nodes are organized in tree-like structures, and each node in the tree monitors its children.

Hierarchical approaches provide high accuracy results with minimum time complexity, but are extremely vulnerable to churn. An alternative approach is based on gossip protocols [1], [6], [7], [10], [13], [14], where information about nodes is exchanged between randomly selected partners and aggregated to produce local estimates.

In the field of gossip-based distribution estimation, Haridasan and van Renesse recently proposed EQUIDEPTH [7]. In this protocol, each node initially divides the set of potential values into fixed-size bins, and over the course of the execution, bins are merged and split based on the number of received values in each bin. In EQUIDEPTH, nodes send the entire current distribution estimate in each message exchange.

Following this research line, Sacha et al. proposed ADAM2 [6], an algorithm that provides an estimation of the cumulative distribution function (CDF) of a given attribute across the population of nodes. The proposed approach allows nodes to compute their own accuracy and to tune the trade-off between communication overhead and estimation accuracy. Similar to EQUIDEPTH, ADAM2 nodes send their whole local vector in each message exchange.

Alternative gossip-based averaging techniques have been proposed by Eyal et al. [5] and Jesus et al. [15], to overcome message loss, network churn and topology changes. In LIMOSENSE [5], each node maintains a pair of values, e.g., a weight and an estimation, that is continuously updated during node communication. Jesus et al. [15] propose a technique where each node uses its current set of neighbors and maintains a dynamic mapping of value flows across to them.

CROUPIER [16], which is a NAT-aware peer sampling service [2], [3], [17], is another system that uses gossip-based estimation. In this system each node has two views, one for *private nodes* behind NATs, and one for *public nodes*. In order to generate a random sample from the two partial views, the

protocol estimates the global ratio of public to private nodes using a gossip-based protocol.

III. System model and problem statement

We consider a network consisting of a collection of *nodes* that communicate through message exchanges. Each node is uniquely identified by a logical ID. The network is highly dynamic and subject to churn, i.e., new nodes may join at any time, and existing nodes may voluntarily leave or crash. We use $\mathcal{N}(t)$ to denote the population of the network at time t. Byzantine behavior is not considered in this work.

We assume that each node in the network has a single local attribute $attr$ that represents a local property, e.g., CPU load or disk space. Let $\mathcal{V}$ be the set of all possible values for $attr$, and let $n(v,t)$ be the number of nodes whose attribute is equal to $v \in V$ at time t. The *global frequency*, $freq(v,t)$, of value v at time t is defined as the fraction of nodes with value v at that time:

$$freq(v,t) = \frac{n(v,t)}{\sum_{w \in \mathcal{V}} n(w,t)} \tag{1}$$

Our goal is to provide each node with an estimate of $freq(v,t)$, for each value v in $\mathcal{V}$, in a completely decentralized way.

IV. The algorithm

Our solution, summarized in Algorithm 1, is based on the gossip paradigm: execution is organized in periodic rounds, performed at roughly the same rate by all nodes, during which a push-pull gossip exchange is executed [18], [19]. During a round, each node p sends a REQUEST message to a partner node q, and waits for the corresponding REPLY message from q. Information contained in the exchanged messages are used to update the local knowledge about the entire system, which is composed by the following information:

- a *partial view* of the network, stored in variable $view$, that represents a small subset of the entire population of nodes;
- a map $count$ that counts the number of times a given value in $\mathcal{V}$ has been received during each of the rounds executed so far.

The partial views are needed to maintain a connected, random overlay topology over the population of all nodes to allow the exchange of information, while $count$ is used to obtain approximate and up-to-date information about the attribute distribution,

Partial views management. Partial views are managed through the CYCLON peer sampling service [2]. We provide here only a brief overview of the protocol, and refer the reader to [2] for a full description. Each partial view contains a fixed number c of *node descriptors* (q,t), composed by a node ID q and a timestamp t. During each round, a node p identifies the node q with the oldest descriptor in its $view$, based on the timestamp through `selectOldest()` in Algorithm 1. The corresponding descriptor is removed, and a subset containing g random descriptors, with $g < c$, is extracted from $view$ through procedure `randomSubset()`. This subset is sent to q through a REQUEST message. The node that receives the REQUEST responses with a REPLY message, that similarly contain g descriptors randomly selected from the local view.

Whenever a subset view $recvView$ is received, procedure `updateView()` merges its content with the node's local view as follows. For each descriptor (q,t) contained in $recvView$, the procedure checks weather q is already included in the local view and its timestamp t' is older than t; if so, the current pair is removed from the local view. Otherwise, if the view has reached its maximum size c, a single random descriptor selected by procedure `randomSubset()` is removed. At this point, there is new space to add (q,t) to the view.

The net effect of this process is the continuous shuffling of views, removing old descriptors belonging to crashed nodes and epidemically disseminating new descriptors generated by active ones. The resulting overlay network, where the neighbors of a node are the nodes included in the partial view, closely resembles a random graph, characterized by extreme robustness and small diameter [2].

Distribution estimation. The map $count$ is indexed by values in $\mathcal{V}$ and by round number, so that $count[v,r]$ counts the number of received messages containing v during round r. We assume that $\mathcal{V}$ is static and known in advance, otherwise, a simple mechanism proposed by Haridasan and van Renesse [7] can adjust the set of entries for the case where the extreme values of a variable are unknown.

At the beginning of round r, $count[v,r]$ is initialized to zero for all values $v \in \mathcal{V}$ and the local attribute value is inserted in the REQUEST message. Whenever an attribute value v is received in round r, $count[v,r]$ is incremented by one.

To estimate the global frequency, we consider a small time window into the past (the *history*), given by the last δ complete rounds. δ is a system parameter that is characterized by a trade-off between the accuracy (the larger δ, the better) and up-to-dateness (the smaller δ, the better) of our estimation. We count in variable $count_\delta[v,r]$ the total number of times that a node has received value v during such period of time:

$$count_\delta[v,r] = \sum_{j=1}^{\delta} count[v, r-j] \tag{2}$$

Our estimate of global frequency of v at round r over the previous δ rounds can thus be computed locally at p as the ratio between the number of received messages by p with value v to the total number of messages received by p:

$$est[v,r] = \frac{count_\delta[v,r]}{\sum_{w \in \mathcal{V}} count_\delta[w,r]} \tag{3}$$

If there is no bias between the average gossip round-time of all nodes and in the message loss between them, $est[v,r]$ can be considered a good approximation of $freq(v,t(r))$, where $t(r)$ is the approximate time when round r has started:

$$est[v,r] \approx freq(v,t(r)) \tag{4}$$

Algorithm 1: Shuffling and estimation algorithm

```
procedure executeRound()
    ▷ Distribution estimation
    round ← round + 1
    computeDistribution(round)
    foreach v ∈ V do
        count[v, round] ← 0
    ▷ Partial view shuffling
    q ← selectOldest(view)
    view.remove(q)
    sub_p ← view.randomSubset(view, g)
    sub_p.add(p, now())
    send ⟨REQUEST, sub_p, attr⟩ to q

on event receive ⟨REQUEST, sub_p, v⟩ from p do
    count[v, round] ← count[v, round] + 1
    sub_q ← view.randomSubset(view, g)
    send ⟨REPLY, sub_q, attr⟩ to p
    view ← updateView(view, sub_p)

on event receive ⟨REPLY, sub_q, v⟩ from q do
    count[v, round] ← count[v, round] + 1
    view ← updateView(view, sub_q)

procedure updateView(view, recvView)
    foreach (q, t) ∈ recvView do
        if (q, t') ∈ view and t' < t then
            view.remove(q)
        else if view.isFull() then
            view.remove(view.randomSubset(1))
        view.add(q, t)

procedure computeDistribution(int r)
    tot ← 0
    foreach v ∈ V do
        count_δ[v, r] ← 0
        for j ← 1 to δ do
            count_δ[v, r] ← count_δ[v, r] + count[v, r − j]
        tot ← tot + count_δ[v, r]
    foreach v ∈ V do
        est[v, r] ← count_δ[v, r]/tot
```

Algorithm 1 shows the details of our protocol. The round code executed periodically is contained in procedure executeRound(), while the code handling messages is shown in the **on event** clauses. Procedure updateView() collects common code that is used when receiving both REQUEST and REPLY messages. Note that this pseudo-code has been designed just to illustrate the main characteristics of the algorithm, and many important optimizations are missing. For example, storing the number of messages received more than δ rounds ago is not necessary, and the current value of $count_\delta$ can be obtained by the previous value by adding the counters of the current round r and removing those of round $r - \delta$.

Improvement. The more values a node receives in a round, the faster it converges to the correct estimate. In the explained model (*baseline* solution), nodes attach only their single local value to each message exchange. However, as we see in Algorithm 1, in each message exchange, together with this value, a small number of node descriptors are sent in sub as well. In the *enhanced* solution, a node descriptor is a triple (q, t, v) composed by a node ID q, a timestamp t and an attribute value v. In this way, a larger number of values are disseminated around and can be used to obtain a more accurate estimate of the distribution in less time. However, we should notice that this improvement is achieved at the cost of a slight increase in the traffic overhead.

V. Experiments

In this section, we evaluate the accuracy of distribution estimation of the baseline and enhanced solutions, and compare them with the existing gossip-based solutions EQUIDEPTH [7] and ADAM2 [6].

We adopt the Kolmogorov-Smirnov (KS) distance [20], to define the upper bound on the approximation error of any node in the system. The KS distance is given by the maximum difference between an estimated distribution and the original distribution. For each node p and for all values $v \in \mathcal{V}$, we measure the *maximum error* at round r and at node p as the distance between $freq(v, t(r))$ and the estimate $est_p[v, r]$, where $t(r)$ is the approximate time when round r has started:

$$maxErr_p(r) = \max_{v \in \mathcal{V}} |freq(v, t(r)) - est_p[v, r]| \quad (5)$$

We measure, then, the *maximum error* at round r as the maximum error over all nodes:

$$maxErr(r) = \max_{p \in \mathcal{N}(r)} maxErr_p(r) \quad (6)$$

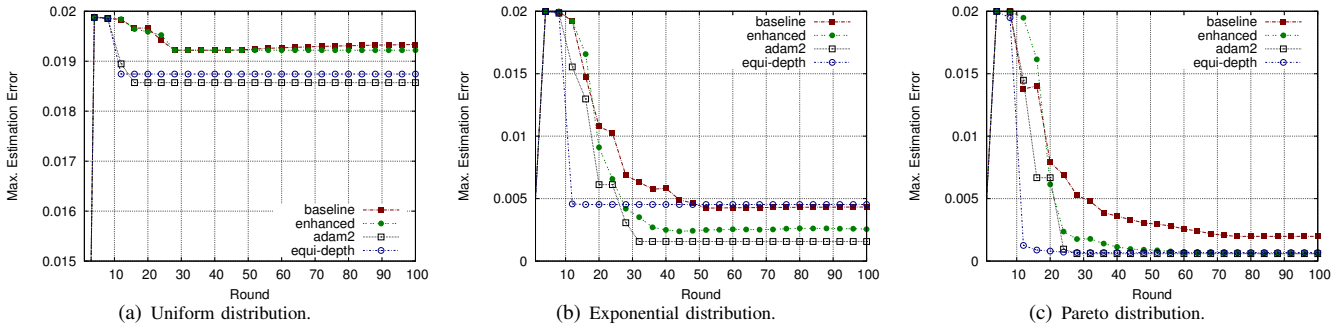
Since the maximum error is determined by a single point difference between $freq(v, t(r))$ and $est_p[v, r]$, it is sensitive to noise, thus, we also measure the average error at each node:

$$avgErr_p(r) = \frac{1}{|\mathcal{V}|} \sum_{v \in \mathcal{V}} |freq(v, t(r)) - est_p[v, r]| \quad (7)$$

Our total average error is then calculated as the average of these local average errors:

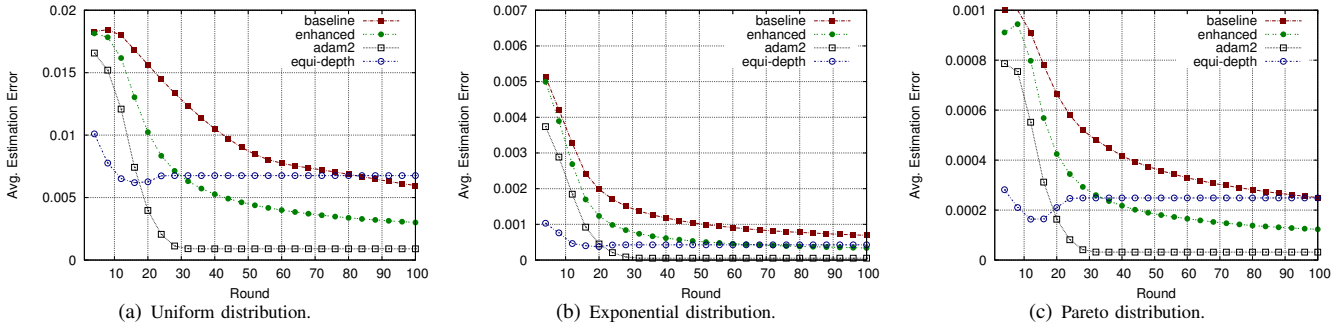
$$avgErr(r) = \frac{1}{|\mathcal{N}(r)|} \sum_{p \in \mathcal{N}(r)} avgErr_p(r) \quad (8)$$

In our experiments, 10,000 nodes participate in the distribution estimation. The nodes join the system following a Poisson distribution with an inter-arrival time of one millisecond. In the experimental setup, for all four protocols, i.e., the baseline model, the enhanced model, EQUIDEPTH and ADAM2, the size of partial view is $c = 10$, and the size of the subset views sent in each view exchange is $g = 5$. The gossiping round period for view exchange is set to one second. Unless stated otherwise, we set the history size δ equal to 100 in both the basic and enhanced model. Latencies between nodes are



(a) Uniform distribution. (b) Exponential distribution. (c) Pareto distribution.

Fig. 1. Maximum estimation error with different distributions.



(a) Uniform distribution. (b) Exponential distribution. (c) Pareto distribution.

Fig. 2. Average estimation error with different distributions.

modeled on Internet latencies, using a latency map based on the King data-set [21].

We consider three value distributions in the experiments: the uniform distribution, the exponential distribution ($\lambda = 1.5$), and the Pareto distribution ($k = 5, x_m = 1$). We assume that the size of $\mathcal{V}$ is equal to 100.

We implemented the set of four protocols in KOMPICS [22], [23], a framework for building P2P protocols that provides a discrete event simulator and several models for bandwidth, latency and churn.

Error. We compare the error and the convergence time of the four solutions in the three test distributions in Figures 1 and 2. As we see ADAM2 converges faster than the other solutions with smaller average error. However, Figure 1 shows that maximum error of the enhanced model and ADAM2 are very close and better than EQUIDEPTH and the baseline. Additionally, we see in Figure 2 that the accuracy of the baseline model increases over time and after a number of rounds its estimation converges to the estimation of EQUIDEPTH.

The local history size. The local history size δ has an important effect on the accuracy of the estimated distribution of the values. If δ equals the system life time, the estimated distribution is approximately equivalent to the real distribution. However, in reality nodes need to bound the size of their history.

Figure 3 shows the average error of the baseline model for different values of δ. In this experiment, the values are distributed uniformly among the nodes. As we see, the bigger δ is, the more accurate the results are.

The traffic overhead. Figure 4 shows the overhead traffic of the four protocols. In this figure the Y-axis shows the cumulative traffic of 10,000 nodes in logarithmic scale. Given that in the baseline model nodes only send their own value, the generated traffic is much smaller than the enhanced model and the two other solutions.

In the enhanced model, the nodes add their values to their descriptors. Therefore, the overhead increases proportionally to the exchanged view size. However, as we see in Figure 4, the enhanced model traffic overhead is much smaller than EQUIDEPTH and ADAM2, which send the whole vector of values.

Churn. Finally, we compare the average error of the four systems in different churn scenarios. In this experiment, we assume three churn rates, such that approximately 0.1%, 1% and 10% of nodes leave the system per second and rejoin immediately as newly initialized nodes [24]. Figure 5 shows that by increasing the churn rate the average error also increases. The figure shows that the enhanced model and ADAM2 compare to EQUIDEPTH have lower average error in 0.1% and 1% churn rates, however, EQUIDEPTH shows a better performance in high churn scenarios.

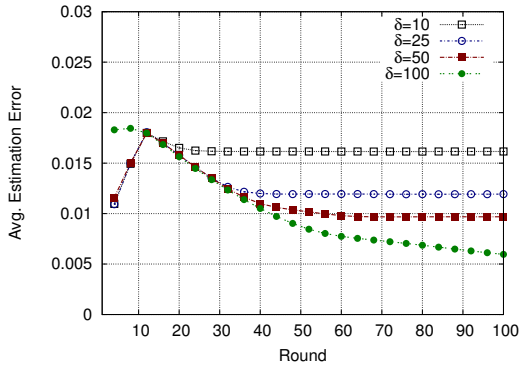


Fig. 3. Accuracy of the baseline solution with different values of δ.

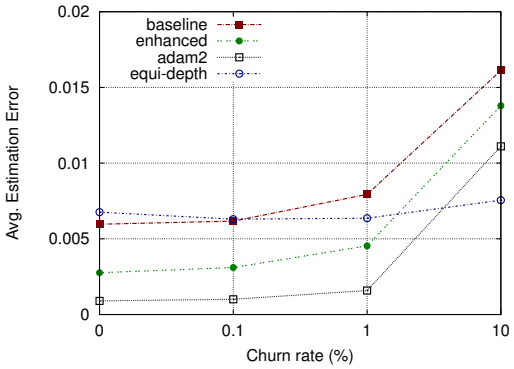


Fig. 5. Average estimation error in different churn rates.

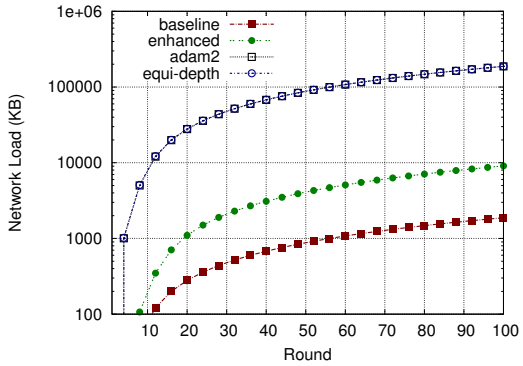


Fig. 4. Traffic overhead.

VI. Conclusions

This paper presents an efficient solution to estimate the distribution of an attribute value, e.g., CPU power or disk space, across all the nodes of a P2P system with high accuracy and low overhead. Our approach is based on the gossip paradigm, where nodes periodically exchange their local information and update it to converge towards a global aggregate value. We thoroughly simulated our proposed algorithm, both for the baseline model and the enhanced one. We compared our protocols with state-of-the-art solutions, like ADAM2 [6] and EQUIDEPTH [7]. We show that maximum error of the enhanced model and ADAM2 are almost the same, and both are smaller than EQUIDEPTH and the baseline model. Moreover, we show that the average error of the enhanced model is also less than EQUIDEPTH, and finally we show that the total network overhead of the baseline model and enhanced model are 1% and 10% those of the EQUIDEPTH and ADAM2, respectively.

References

[1] M. Jelasity, A. Montresor, and O. Babaoglu, "Gossip-based aggregation in large dynamic networks," *ACM Trans. Comput. Syst.*, vol. 23, no. 3, pp. 219–252, 2005.

[2] S. Voulgaris, D. Gavidia, and M. van Steen, "CYCLON: Inexpensive Membership Management for Unstructured P2P Overlays," *Journal of Network and Systems Management*, vol. 13, no. 2, pp. 197–217, 2005.

[3] A. Payberah, J. Dowling, and S. Haridi, "Gozar: Nat-friendly peer sampling with one-hop distributed nat traversal," in *Proc. of DAIS'11*. Springer, pp. 1–14.

[4] A. Payberah, H. Kavalionak, V. Kumaresan, A. Montresor, and S. Haridi, "CLive: Cloud-assisted P2P live streaming," in *Proc. of P2P'12*. IEEE, pp. 79–90.

[5] I. Eyal, I. Keidar, and R. Rom, "Limosense – live monitoring in dynamic sensor networks," in *Proc. of LiMoSense'10*. Springer, pp. 72–85.

[6] J. Sacha, J. Napper, C. Stratan, and G. Pierre, "Adam2: Reliable distribution estimation in decentralised environments," in *Proc. of ICDCS'10*. IEEE, pp. 697–707.

[7] M. Haridasan and R. Van Renesse, "Gossip-based distribution estimation in peer-to-peer networks," in *Proc. of IPTPS'08*. USENIX Association, pp. 13–13.

[8] R. Van Renesse, K. Birman, and W. Vogels, "Astrolabe: A robust and scalable technology for distributed system monitoring, management, and data mining," *ACM Trans. Comput. Syst.*, vol. 21, pp. 164–206, 2003.

[9] P. Yalagandula and M. Dahlin, "A scalable distributed information management system," in *Proc. of SIGCOMM'04*. ACM, pp. 379–390.

[10] A. Montresor, M. Jelasity, and O. Babaoglu, "Decentralized ranking in large-scale overlay networks," in *Proc. of SASOW'08*. IEEE, pp. 208–213.

[11] K. Graffi, A. Kovacevic, S. Xiao, and R. Steinmetz, "Skyeye.kom: An information management over-overlay for getting the oracle view on structured P2P systems," in *Proc. of ICPADS'08*. IEEE, pp. 279–286.

[12] D. Jurca and R. Stadler, "H-gap: estimating histograms of local variables with accuracy objectives for distributed real-time monitoring," *IEEE Tran. Net. Ser. Man.*, vol. 7, no. 2, pp. 83–95, 2010.

[13] R. Van Renesse, Y. Minsky, and M. Hayden, "A gossip-style failure detection service," in *Proc. of Middleware'98*. Springer.

[14] S. Idreos, M. Koubarakis, and C. Tryfonopoulos, "P2P-DIET: an extensible P2P service that unifies ad-hoc and continuous querying in super-peer networks," in *Proc. of SIGMOD'04*. ACM, pp. 933–934.

[15] P. Jesus, C. Baquero, and P. Almeida, "Fault-tolerant aggregation for dynamic networks," in *Proc. of SRDS'10*. IEEE, pp. 37–43.

[16] J. Dowling and A. Payberah, "Shuffling with a croupier: Nat-aware peer-sampling," in *Proc. of ICDCS'12*. IEEE, pp. 102–111.

[17] M. Jelasity and A. Montresor, "Epidemic-style proactive aggregation in large overlay networks," in *Proc. of ICDCS'04*. IEEE, pp. 102–109.

[18] M. Jelasity, S. Voulgaris, R. Guerraoui, A. Kermarrec, and M. Van Steen, "Gossip-based peer sampling," *ACM Trans. Comput. Syst.*, vol. 25, no. 3, 2007.

[19] A. Demers *et al.*, "Epidemic algorithms for replicated database maintenance," *SIGOPS Oper. Syst. Rev.*, vol. 22, no. 1, pp. 8–32, 1988.

[20] G. Schay, *Introduction to probability with statistical applications*. Birkhäuser, 2007.

[21] K. Gummadi, S. Saroiu, and S. Gribble, "King: Estimating latency between arbitrary internet end hosts," *SIGCOMM Comput. Commun. Rev.*, vol. 32, no. 3, pp. 11–11, 2002.

[22] C. Arad, J. Dowling, and S. Haridi, "Developing, simulating, and deploying peer-to-peer systems using the Kompics component model," in *Proc. of COMSWARE'09*. ACM, pp. 16:1–16:9.

[23] ——, "Message-passing concurrency for scalable, stateful, reconfigurable middleware," in *Proc. of Middleware'12*. Springer, pp. 208–228.

[24] D. Stutzbach and R. Rejaie, "Understanding churn in peer-to-peer networks," in *Proc. of IMC'06*. ACM, pp. 189–202.